Lớp:	CE224.P15.1
Tên thành viên:	Trần Quang Huy – 22520578 Nguyễn Lê Thanh Hiển – 22520418 Nguyễn Hoàng Tùng – 22521618 Đoàn Vũ Phú Minh – 22520859 Nguyễn Xuân Lộc - 22520793

MINI PROJECT:
SINGLE PING-PONG GAME

1 CHUẨN BỊ

1.1 Phần cứng

- KIT STM32F4 Discovery for STM32F429 MCU.

1.2 Phần mềm

- STM32CubeIDE: sử dụng để lập trình, build, nạp và debug code.

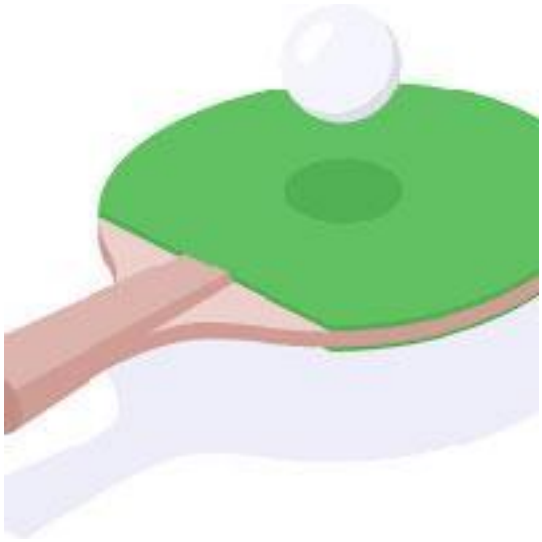
1.3 Kiến thức

- Biết cách tạo project và cấu hình project sử dụng STM32CubeIDE
- Có kiến thức vững về Hệ điều hành, đặc biệt là vấn đề lập lịch và đồng bộ các tiến trình

Hình 1.1. KIT STM32F4 Discovery

2 MÔ TẢ

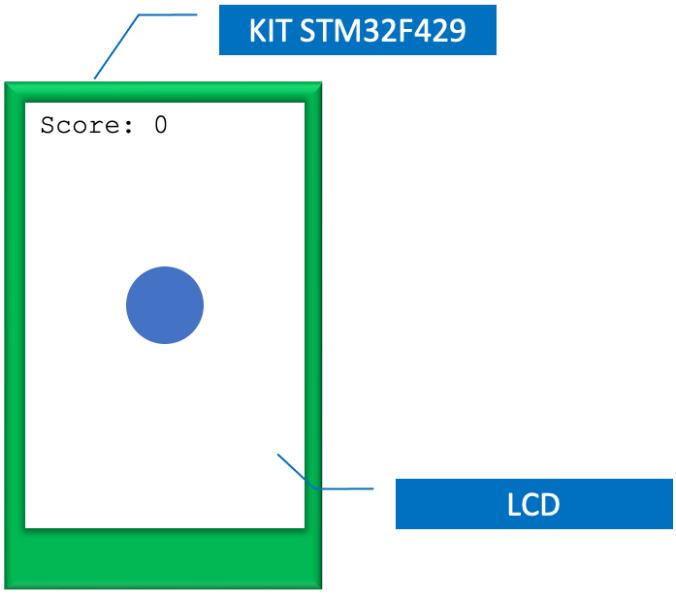
Single Ping-Pong Game là một trò chơi mô phỏng việc dùng vợt và tâng bóng bàn được mô tả như trong Hình 1, trong đó, KIT STM32F429 được sử dụng như cây vợt, màn hình LCD sẽ hiển thị một hình tròn mô phỏng trái bóng bàn.



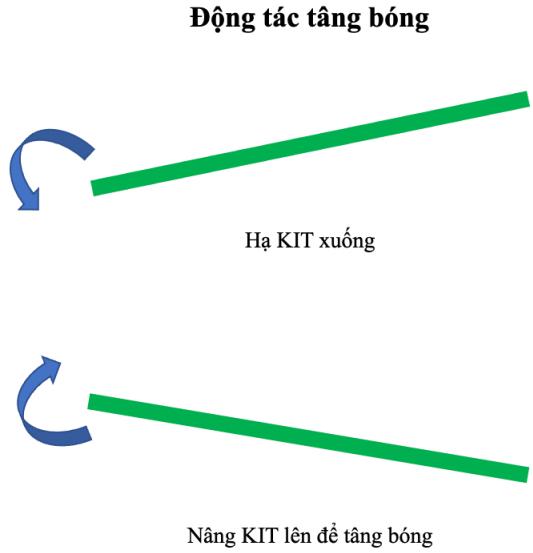
Hình 1. Mô tả game tâng bóng bàn

Nguyên lý của game như sau:

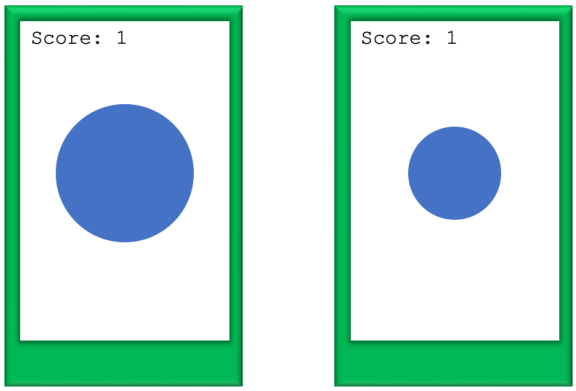
- Đầu tiên, hệ thống ở trạng thái cân bằng khởi đầu.
- Người chơi sẽ hạ và nâng KIT để thực hiện động tác tâng bóng, khi đó trên màn hình LCD sẽ mô phỏng trạng thái trái bóng được nâng lên và rơi xuống.
- Mục tiêu của người chơi là phải nâng KIT đúng lúc trái bóng rơi xuống, đập vào "mặt vợt" và tâng lên lại, người chơi được tính điểm, đèn xanh chớp 1 lần.
- Nếu người chơi nâng đúng, tùy thuộc vào mức độ chính xác mà bóng sẽ nảy lên với tốc độ khác nhau.
- Nếu người tâng "hụt" thì bóng sẽ rớt và GAME OVER, đèn đỏ được bật và giữ cho tới khi reset.



Hình 2. Trạng thái bắt đầu của game



Hình 3. Mô phỏng động tác tâng bóng



Mô phỏng bóng bay lên Mô phỏng bóng rớt xuống

Hình 4. Mô phỏng bóng được nâng và rớt

Yêu cầu về mức độ hoàn thành:

Yêu cầu	Điểm
Hiển thị được giao diện ban đầu	2 điểm
Hiện thực được thao tác tâng bóng tại chỗ (điểm giữa màn hình)	3 điểm
Xuất thông số độ cao khi bóng được tâng ra máy tính qua Virtual Com Port	2 điểm
Hiện thực được chức năng tính điểm	3 điểm
Hiện thực được việc bóng bay theo 1 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	1 điểm
Hiện thực được việc bóng bay theo 2 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	1 điểm

3 BÀI TẬP

Bài tập 1: Thiết kế mô hình phần cứng cứng thiết cho yêu cầu trên: cần dùng những phần cứng nào, sử dụng các giao thức giao tiếp gì giữa các phần cứng? **(15% số điểm)**

- Các phần cứng cần sử dụng: STM32F429, gyroscope, LCD, Virtual Comport, GPIO

- Cách giao tiếp giữa các phần cứng:

+ Gyroscope: dùng thư viện hỗ trợ (Board Support Package - BSP) và giao tiếp bằng giao thức SPI

+ LCD: dùng thư viện hỗ trợ (Board Support Package - BSP)

+ Virtual Comport: USB OTG HS với thư viện hỗ trợ usbd_cdc_if.h

+ GPIO: sử dụng các led on-board và nút reset

Bài tập 2: Sử dụng RTOS, thiết mô hình phần mềm cho yêu cầu trên: nêu công việc của từng task, luồng xử lý dữ liệu như thế nào? **(15% số điểm)**

- Chương trình có 4 tasks chính:

Task01: Đọc giá trị từ gyroscope

- Đọc giá trị từ cảm biến gyroscope thông qua hàm L3GD20_ReadXYZAngRate.
- Xử lý giá trị trục X:
  - Nếu vượt ngưỡng, đặt hitflag = 1 và gửi dữ liệu vào hàng đợi Queue01.
  - Nếu không, đặt hitflag = 0.

Task02: Hiển thị và kiểm soát bóng trên LCD

- Quản lý bóng trên màn hình LCD.
- Khi hitflag = 1:
  1. Nhận giá trị lực từ hàng đợi.
  2. Tăng bán kính của bóng (tăng chiều cao).
  3. Nếu bán kính đạt ngưỡng, tăng điểm số (realScore) và bật LED báo hiệu (GPIOG_PIN_13).
  4. Giảm bán kính khi bóng rơi xuống.

5. Nếu bóng rơi xuống dưới ngưỡng (radius < 20), kết thúc trò chơi (flag = 1).

Task03: Gửi dữ liệu qua USB (CDC)

- Gửi giá trị bán kính (radius) của bóng qua serial với tốc độ 100ms/lần.

Task04: Hiển thị điểm số và trạng thái trò chơi

- Hiển thị điểm số hiện tại trên LCD khi flag = 0.
- Khi flag = 1 (kết thúc trò chơi):
  - Hiển thị thông báo "GAME OVER".
  - Bật LED GPIOG_PIN_14 để báo hiệu.
  - Dừng các task khác bằng vTaskDelete.

- Luồng xử lý dữ liệu:

1. **Khởi tạo**:
   - Khởi tạo các biến, LCD, gyroscope, GPIO, và FreeRTOS.

```
2. /* Includes ------------------------------------------------------------
   ---------*/
3. #include "main.h"
4. #include "cmsis_os.h"
5. #include "usb_device.h"
6.
7. /* Private includes ----------------------------------------------------
   ---------*/
8. /* USER CODE BEGIN Includes */
9. #include "usbd_cdc_if.h"
10.      #include "stm32f429i_discovery_sdram.h"
11.      #include "stm32f429i_discovery_lcd.h"
12.      #include "stm32f429i_discovery_gyroscope.h"
13.      /* USER CODE END Includes */
14.
15.      /* Private typedef -----------------------------------------------
   ---------------*/
16.      /* USER CODE BEGIN PTD */
17.      typedef struct{
18.           int Buf[6];
19.      } Mymessage;
20.      /* USER CODE END PTD */
21.
22.      /* Private define ------------------------------------------------
   ---------------*/
23.      /* USER CODE BEGIN PD */
24.      uint8_t buf1[10];
25.      uint8_t buf2[10];
```

```
26.       int realScore = 0;       //điểm
27.       int flag = 0;            //cờ kết thúc
28.       int radius = 20;  //bán kính, chiều cao của hình tròn
29.       int hitflag = 0;  //cờ báo khi đánh trúng bóng
30.       float gyroValue[3];      //mảng gyroscope
31.       int X_circle = 115;
32.       int Y_circle = 150;
33.       /* USER CODE END PD */
34.
35.       /* Private macro -------------------------------------------------
    ---------------*/
36.       /* USER CODE BEGIN PM */
37.
38.       /* USER CODE END PM */
39.
40.       /* Private variables ---------------------------------------------
    ---------------*/
41.       /* Definitions for Task01 */
42.       osThreadId_t Task01Handle;
43.       const osThreadAttr_t Task01_attributes = {
44.         .name = "Task01",
45.         .stack_size = 128 * 4,
46.         .priority = (osPriority_t) osPriorityNormal,
47.       };
48.       /* Definitions for Task02 */
49.       osThreadId_t Task02Handle;
50.       const osThreadAttr_t Task02_attributes = {
51.         .name = "Task02",
52.         .stack_size = 128 * 4,
53.         .priority = (osPriority_t) osPriorityNormal,
54.       };
55.       /* Definitions for Task03 */
56.       osThreadId_t Task03Handle;
57.       const osThreadAttr_t Task03_attributes = {
58.         .name = "Task03",
59.         .stack_size = 128 * 4,
60.         .priority = (osPriority_t) osPriorityNormal,
61.       };
62.       /* Definitions for Task04 */
63.       osThreadId_t Task04Handle;
64.       const osThreadAttr_t Task04_attributes = {
65.         .name = "Task04",
66.         .stack_size = 128 * 4,
67.         .priority = (osPriority_t) osPriorityNormal,
68.       };
69.       /* Definitions for Queue01 */
70.       osMessageQueueId_t Queue01Handle;
71.       const osMessageQueueAttr_t Queue01_attributes = {
72.         .name = "Queue01"
73.       };
74.       /* USER CODE BEGIN PV */
75.
76.       /* USER CODE END PV */
```

```
77.
78.       /* Private function prototypes -----------------------------------
   ---------------*/
79.       void SystemClock_Config(void);
80.       static void MX_GPIO_Init(void);
81.       void StartTask01(void *argument);
82.       void StartTask02(void *argument);
83.       void StartTask03(void *argument);
84.       void StartTask04(void *argument);
```

2. Vòng lặp chính (scheduler): Khởi tạo hàng đợi Queue01 và các Task

```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
---*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  BSP_SDRAM_Init();
  MX_USB_DEVICE_Init();
  BSP_GYRO_Init();
  BSP_LCD_Init();//init LCD
  BSP_LCD_LayerDefaultInit(1, SDRAM_DEVICE_ADDR);//set the layer buffer
address into SDRAM
  BSP_LCD_SelectLayer(1);//select on which layer we write
  BSP_LCD_DisplayOn();//turn on LCD
  BSP_LCD_Clear(LCD_COLOR_BLUE);//clear the LCD on blue color
  BSP_LCD_SetBackColor(LCD_COLOR_BLUE);//set text background color
  BSP_LCD_SetTextColor(LCD_COLOR_WHITE);//set text color
  /* USER CODE END 2 */
```

```
  /* Init scheduler */
  osKernelInitialize();

  /* USER CODE BEGIN RTOS_MUTEX */
  /* add mutexes, ... */
  /* USER CODE END RTOS_MUTEX */

  /* USER CODE BEGIN RTOS_SEMAPHORES */
  /* add semaphores, ... */
  /* USER CODE END RTOS_SEMAPHORES */

  /* USER CODE BEGIN RTOS_TIMERS */
  /* start timers, add new ones, ... */
  /* USER CODE END RTOS_TIMERS */

  /* Create the queue(s) */
  /* creation of Queue01 */
  Queue01Handle = osMessageQueueNew (16, sizeof(Mymessage),
&Queue01_attributes);

  /* USER CODE BEGIN RTOS_QUEUES */
  /* add queues, ... */
  /* USER CODE END RTOS_QUEUES */

  /* Create the thread(s) */
  /* creation of Task01 */
  Task01Handle = osThreadNew(StartTask01, NULL, &Task01_attributes);

  /* creation of Task02 */
  Task02Handle = osThreadNew(StartTask02, NULL, &Task02_attributes);

  /* creation of Task03 */
  Task03Handle = osThreadNew(StartTask03, NULL, &Task03_attributes);

  /* creation of Task04 */
  Task04Handle = osThreadNew(StartTask04, NULL, &Task04_attributes);

  /* USER CODE BEGIN RTOS_THREADS */
  /* add threads, ... */
  /* USER CODE END RTOS_THREADS */

  /* USER CODE BEGIN RTOS_EVENTS */
  /* add events, ... */
  /* USER CODE END RTOS_EVENTS */

  /* Start scheduler */
  osKernelStart();
```

- Task01 đọc giá trị từ gyroscope: Sử dụng hàm L3GD20_ReadXYZAngRate(gyroValue) để đọc giá trị gia tốc góc sau đó kiểm tra nếu đánh trúng bóng (hitflag = 1) thì gửi dữ liệu vào Queue

```
85.       void StartTask01(void *argument)
86.       {
87.
88.         /* USER CODE BEGIN 5 */
89.               Mymessage SendX;
90.               /* Infinite loop */
91.               for(;;)
92.               {
93.                   if(flag == 0)
94.                   {
95.                   L3GD20_ReadXYZAngRate(gyroValue);
96.                   SendX.Buf[0] = (int)gyroValue[0]/1500;
97.                   SendX.Buf[1] = (int)gyroValue[1];
98.                   SendX.Buf[2] = (int)gyroValue[2];
99.                   if (abs(SendX.Buf[0]) > 20)
100.                  {
101.                        hitflag = 1;
102.                        osMessageQueuePut(Queue01Handle, &SendX, 0,
    0);
103.                  }
104.                  else
105.                  {
106.                        hitflag = 0;
107.                  }
108.                osDelay(250);
109.                  }
110.              }
111.        /* USER CODE END 5 */
112.      }
```

- Task02 xử lý bóng và cập nhật điểm số:

- Vẽ quả bóng lên màn hình LCD (sử dụng hàm BSP_LCD_FillCircle).

- Nếu hitflag == 1 và trò chơi chưa kết thúc (flag == 0), nhận dữ liệu từ hàng đợi Queue01:

1. Quá trình bóng bay lên:
   - Bóng được "nâng cao" bằng cách tăng bán kính dần (radius).
   - Nếu bán kính đạt 50, cộng điểm (realScore) và bật đèn LED (GPIO pin 13) báo hiệu.

2. Quá trình bóng rơi xuống:
    - Nếu không đủ lực, bóng sẽ rơi xuống với bán kính giảm dần.
    - Nếu bán kính nhỏ hơn 20, trò chơi kết thúc (flag = 1).

```
113.      void StartTask02(void *argument)
114.      {
115.        /* USER CODE BEGIN StartTask02 */
116.                  int temp;
117.                  BSP_LCD_FillCircle(X_circle, Y_circle, radius);
118.                  Mymessage GetX;
119.                  for(;;)
120.                  {
121.                        if (hitflag == 1 && flag == 0)
122.                        {
123.                              osMessageQueueGet(Queue01Handle, &GetX,
    0, 0);
124.                              temp = GetX.Buf[0];
125.                              for (int i = 0; i <= temp; i = i + 10)
126.                              {
127.                                    BSP_LCD_Clear(LCD_COLOR_BLUE);
128.                                    BSP_LCD_FillCircle(X_circle,
    Y_circle, radius);
129.                                    radius = radius + 5;
130.                                    if(radius==50){
131.                                          realScore = realScore + 1;
132.                                          HAL_GPIO_WritePin(GPIOG,
    GPIO_PIN_13, GPIO_PIN_SET);
133.                                          osDelay(250);
134.                                          HAL_GPIO_WritePin(GPIOG,
    GPIO_PIN_13, GPIO_PIN_RESET);
135.                                    }
136.                                    if (radius > 90){
137.                                          break;
138.                                    }
139.                                    osDelay(100);
140.                              }
141.                              if(temp<=140)//Set lại giá trị temp khi
    tăng bóng không đủ lực
142.                                    temp = 200;
143.                              for (int i = temp; i >= 0; i = i - 10)
144.                              {
145.                                    BSP_LCD_Clear(LCD_COLOR_BLUE);
146.                                    BSP_LCD_FillCircle(X_circle,
    Y_circle, radius);
147.                                    radius = radius - 5;
148.                                    if (radius >= 20 && radius <= 50)
149.                                    {
150.                                          if(hitflag==1)
151.                                                break;
```

```
152.                                    }
153.                                    if (radius < 20)
154.                                    {
155.                                          flag = 1;
156.                                          break;
157.                                    }
158.                                    osDelay(100);
159.                              }
160.                        }
161.                  }
162.       /* USER CODE END StartTask02 */
163.     }
```

- Task03 truyền dữ liệu qua USB: sử dụng hàm CDC_Transmit_HS để gửi dữ liệu qua serial mỗi 100ms

```
164.     void StartTask03(void *argument)
165.     {
166.       /* USER CODE BEGIN StartTask03 */
167.                 for(;;)
168.                 {
169.                       sprintf(buf2, "\n%d", radius);
170.                       CDC_Transmit_HS(buf2,sizeof(buf2));
171.                       osDelay(100);
172.                 }
173.       /* USER CODE END StartTask03 */
174.     }
```

- Task04 hiển thị thông tin lên LCD và kiểm tra trạng thái trò chơi:

- Nếu trò chơi đang diễn ra (flag == 0):

  - Hiển thị điểm số hiện tại (realScore) trên LCD (sử dụng BSP_LCD_DisplayStringAtLine).

- Nếu trò chơi kết thúc (flag == 1):

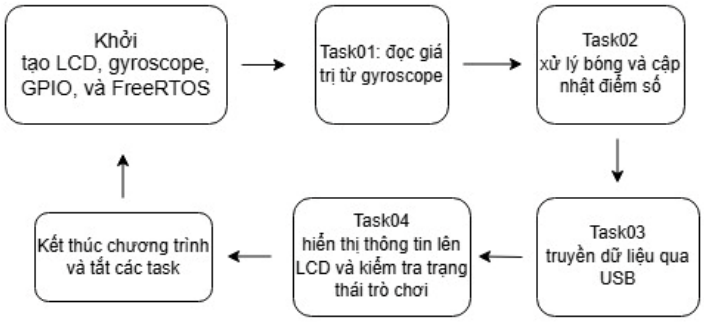
  - Xóa màn hình LCD và chuyển sang màu đỏ.
  - Hiển thị dòng chữ "GAME OVER" cùng điểm số cuối cùng.
  - Bật đèn LED (GPIO pin 14) để báo hiệu.
  - Dừng các task Task01, Task02, và Task03 bằng cách gọi vTaskDelete

```
175.      void StartTask04(void *argument)
176.      {
177.        /* USER CODE BEGIN StartTask04 */
178.        /* Infinite loop */
179.        for(;;)
180.        {
181.                  if (flag == 0)
182.                  {
183.                        sprintf((char*)buf1, "Score: %d", realScore);
184.                        BSP_LCD_DisplayStringAtLine(1, buf1);
185.                  }
186.                  else if (flag == 1)
187.                  {
188.                        BSP_LCD_Clear(LCD_COLOR_RED);
189.                        BSP_LCD_SetBackColor(LCD_COLOR_RED);
190.                        BSP_LCD_DisplayStringAt(1,130,(uint8_t*)"GAME
   OVER", CENTER_MODE);
191.
         BSP_LCD_DisplayStringAt(1,150,(uint8_t*)buf1,CENTER_MODE);
192.                        HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14,
   GPIO_PIN_SET);
193.                        vTaskDelete(Task01Handle);
194.                        vTaskDelete(Task02Handle);
195.                        vTaskDelete(Task03Handle);
196.                  }
197.        }
198.        /* USER CODE END StartTask04 */
199.      }
```

3. Kết thúc trò chơi:

- Hiển thị "GAME OVER".
- Tắt các task và dừng chương trình

Bài tập 3: Thiết kế lưu đồ giải thuật xử lý cho yêu cầu trên? **(30% số điểm)**



Bài tập 4: Hiện thực hệ thống và báo cáo kết quả? **(40% số điểm)**

Link video:

https://drive.google.com/file/d/11hRqvjUy8KRDUVvoxMTw31UuQKCaY3_8/view?usp=sharing

Tham khảo:

https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__Message.html

https://hocarm.org/rtos-co-ban-phan-1/

https://hocarm.org/rtos-co-ban-phan-2/

https://www.st.com/resource/en/datasheet/l3gd20.pdf

https://stm32f4-discovery.net/2014/08/library-28-l3gd20-3-axis-gyroscope/

https://itecnotes.com/electrical/converting-raw-gyro-l3gd20h-values-into-angles/